

Facultad de Ingeniería de Sistemas Computacionales

Capítulo II. Elementos básicos de un algoritmo

Kexy Rodríguez

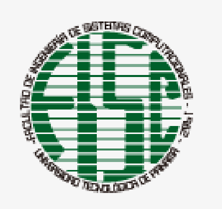




Contenido

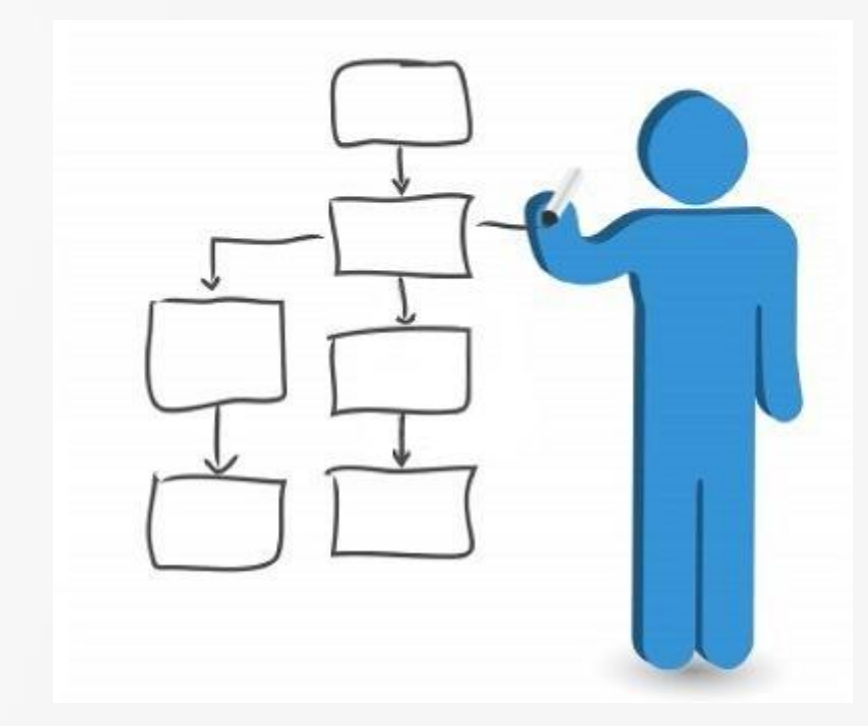
- Estructura de un algoritmo en Pseudocódigo
- Reglas de escritura de un Algoritmo en pseudocódigo
- Elementos Básicos
- Identificadores
- Variables / Constantes
- Tipos de Datos
- Operadores Aritméticos, Expresiones y Jerarquía
- Asignación
- Entrada / Salida de datos en pseudocódigo





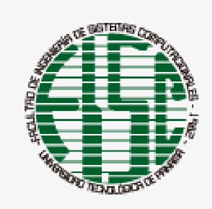
Estructura de un algoritmo en Pseudocódigo

La estructura de un algoritmo nos funciona para tener una mejor organización de los elementos el cual está definida en tres partes.



- ✓**Cabecera.**
- ✓**Declaraciones.**
- ✓**Cuerpo.**





Estructura de un algoritmo en Pseudocódigo

Cabecera:

Se debe indicar el nombre –identificador– asignado al algoritmo

algoritmo <nombre_del_algoritmo>

Ejemplos:

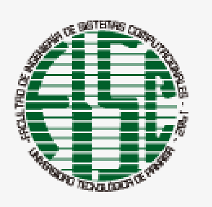
algoritmo Calculo_de_impuesto

algoritmo Area_de_una_circunferencia

algoritmo Ingresar_a_la_UTP

algoritmo Participar_en_la_JIC





Estructura de un algoritmo en Pseudocódigo

Declaraciones:

En esta sección se declaran las constantes, los tipos de datos y las variables

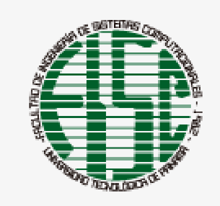
algoritmo <nombre_del_algoritmo>

[**constantes**
<declaraciones_de_constantes>]

[**tipos_de_datos**
<declaraciones_de_tipos_de_datos>]

[**variables**
<declaraciones_de_variables>]





Estructura de un algoritmo en Pseudocódigo

Declaraciones:

Ejemplo:

algoritmo Calculo_de_impuesto

  constantes

    impuesto ← 0.07

  variables

    real precio





Estructura de un algoritmo en Pseudocódigo

Cuerpo:

En esta sección se escribe todas las instrucciones del algoritmo

INICIO

<instrucción_1>

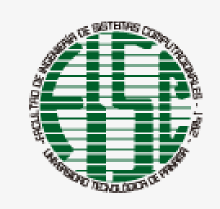
<instrucción_2>

...

<instrucción_n>

FIN





Estructura de un algoritmo en Pseudocódigo

Algoritmo Calculo_de_impuesto

constantes

impuesto ← 0.07

variables

real precio, calculoDeImpuesto

INICIO

imprimir("Introduzca el precio del producto: ")

leer(precio)

calculoDeImpuesto ← precio * impuesto

imprimir(“El impuesto del producto es: “, calculoDeImpuesto)

FIN





Práctica

- Escribir un algoritmo donde transforma de libras a kilogramos
- Escribir un algoritmo donde se obtiene el promedio de 5 calificaciones
- Escribir un algoritmo donde pregunte el nombre apellido y edad y diga si es mayor de edad





Buenas prácticas

1.

Organización

- Estructuración

2.

Orden

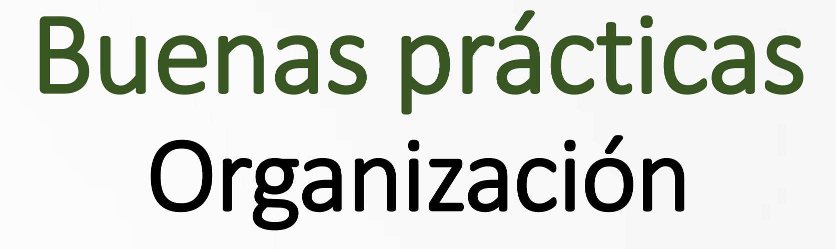
- Escritura
- Documentación

3.

Control

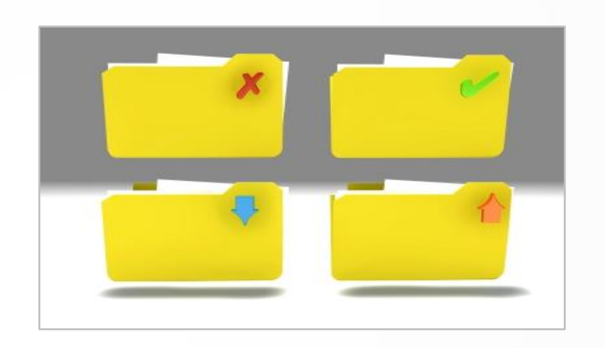
- Control de versiones

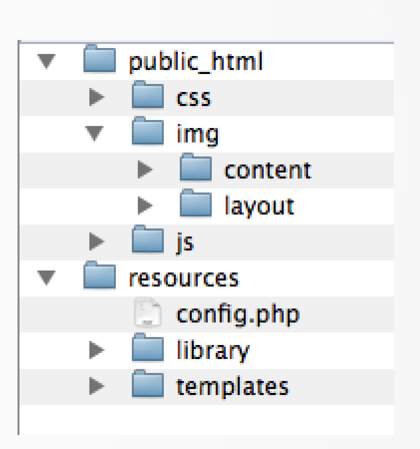


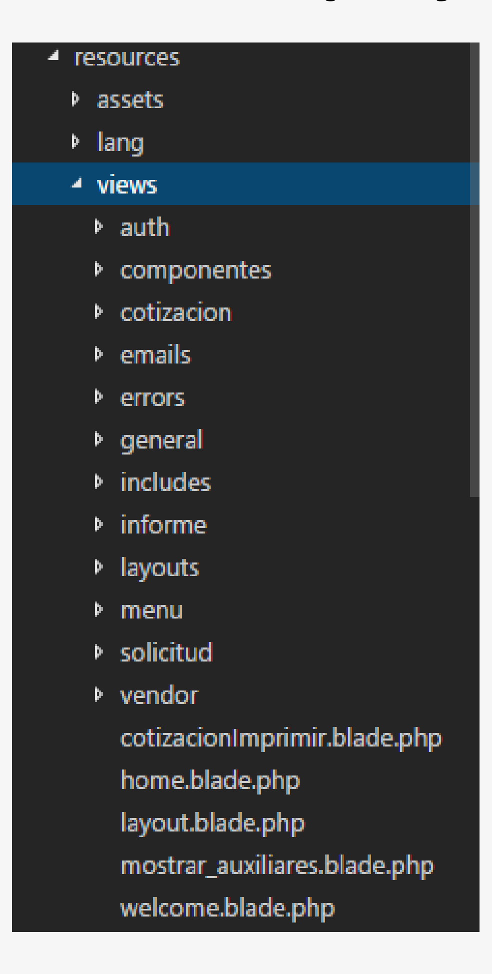




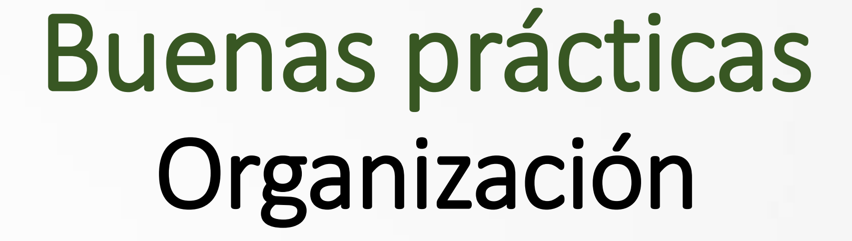
- **Estructuración - Organización de los directorios del proyecto**









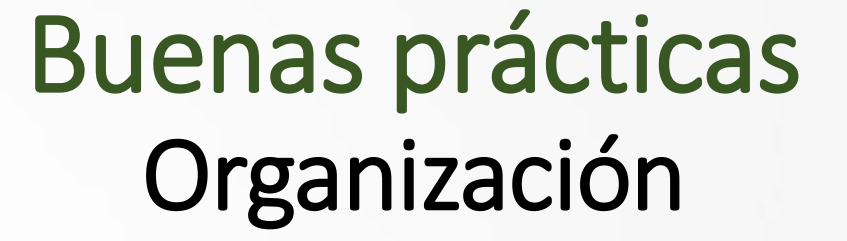


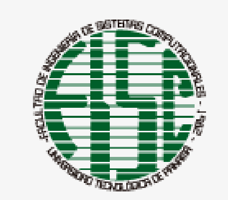


- **Estructuración**

Indentación	Refactorización







- **Estructuración - Indentación**

La indentación es un tipo de notación secundaria utilizado para **mejorar la legibilidad del código** fuente por parte de los programadores.

En ciertos lenguajes de programación como Haskell, Occam y Python, el sangrado se utiliza para delimitar la estructura del programa permitiendo establecer bloques de código.

```
public void conIndentacion(){
    int i, j;
    for (i = 0; i <= 10; i++){
        for (j = 0; j <= 10; j++){
        System.out.print("%i x %i = %i\n"+ i + j + i * j);
        }
    }
}

public void sinIndentacion(){
int i, j;
for (i = 0; i <= 10; i++){
for (j = 0; j <= 10; j++){
System.out.print("%i x %i = %i\n"+ i + j + i * j);
}
}
}
```

(Francisco Marchena, 2017)



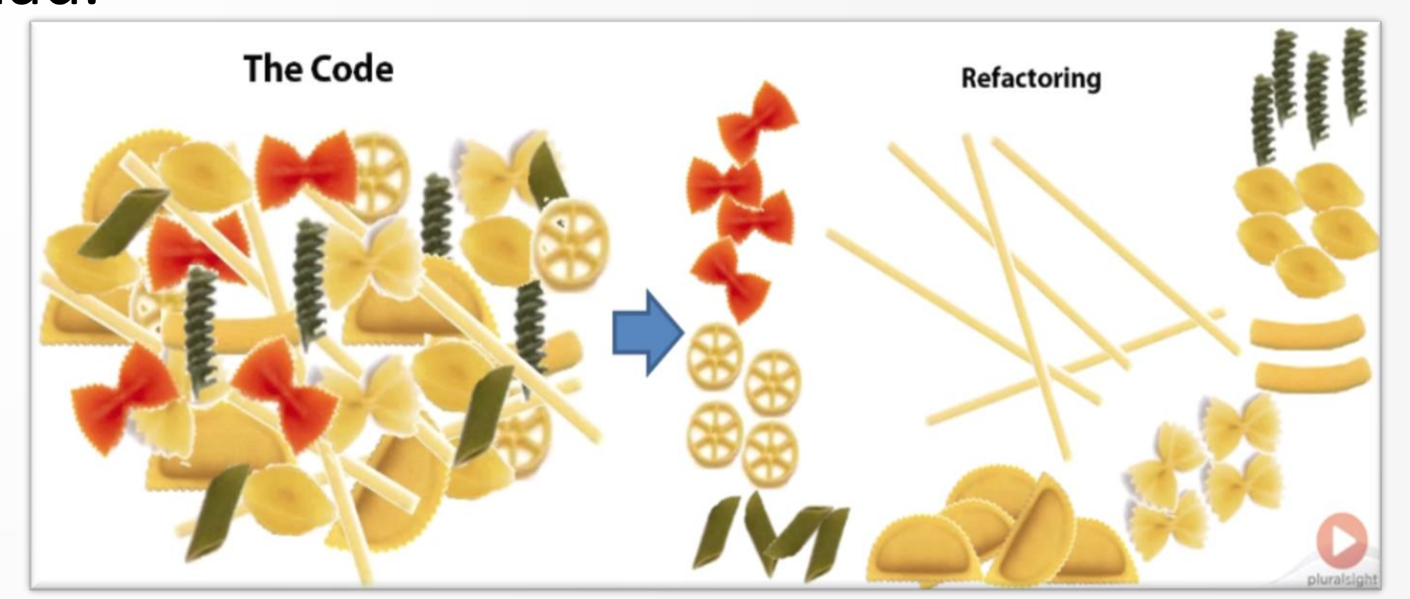
Buenas prácticas

Organización



- **Estructuración - Refactorización**

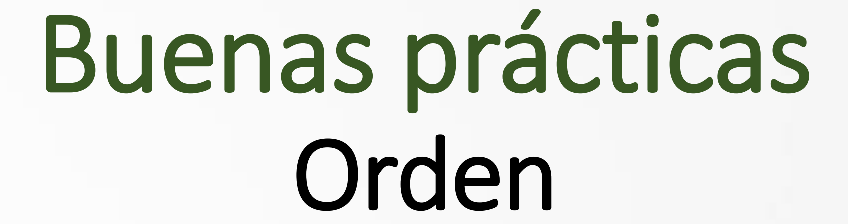
Es una técnica de la ingeniería de software para reestructurar un código fuente, alterando su estructura interna sin cambiar su comportamiento externo. La refactorización es la parte del mantenimiento del código que no arregla errores ni añade funcionalidad.



(Francisco Marchena, 2017)



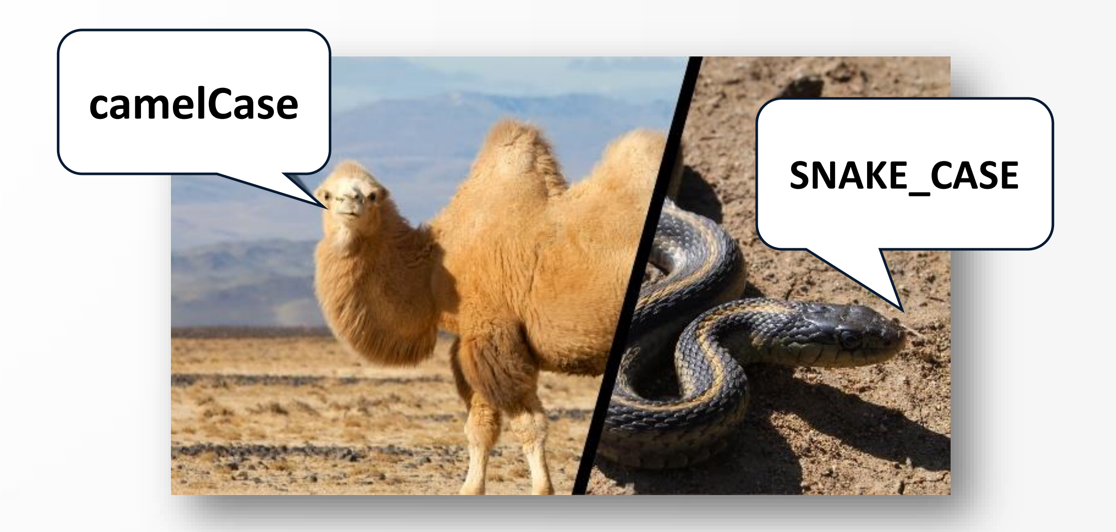




Buenas prácticas
Orden

- **Escritura**

camelCase	SNAKE_CASE	PascalCase









Buenas prácticas

Orden

- **Escritura**

Variables

- **camelCase**: se utiliza para nombrar **variables de trabajo**. Se coloca la primera letra en minúscula y la siguientes palabras la primera letra en mayúscula.

```
if($val==1) {
    $tmpName = "";
    $nombreFile = "";
    $directorio = "informe";
```

Constantes

- **SNAKE_CASE**: utilizada para nombrar **constantes**. Se coloca todo en mayúscula cerrada separado por guion abajo (_).

```
=> 'pgsql',
=> env('DB_HOST', 'localhost'),
=> env('DB_DATABASE', 'forge'),
=> env('DB_USERNAME', 'forge'),
=> env('DB_PASSWORD', ''),
=> 'utf8',
```

(Francisco Marchena, 2017)





Buenas prácticas
Orden



- **Escritura**

  - **PascalCase**: se utiliza esta notación para nombrar las **clases, los módulos y funciones**.

  Todas las primeras letras van en mayúscula, ayuda a que se haga la interpretación de que estamos en un archivo o función importante

```
class GetFilesController extends Controller
{

    protected  $dir=DIRECTORY_SEPARATOR;
    /**
    * Display a listing of the resource.
    *
    * @return \Illuminate\Http\Response
    */

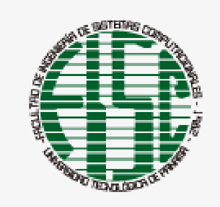
+   public function GetFilePlantilla($directorio,$filename,$noExtension
+   protected function GetExtenFilePlantilla($directorio, $fileName)…
   /**
   * Store a newly created resource in storage.
```

(Francisco Marchena, 2017)



Buenas prácticas

Orden



- Otras reglas básicas de escritura de código son:
  - ✓ Los **nombres de las funciones** deben ser **verbos y empezar en mayúscula**.
  - ✓ El **nombre del objeto es el mismo que la clase** aplicando camelCase.
  - ✓ Si el lenguaje lo permite, **separa los valores de los operadores**.
  - ✓ Recuerda siempre **inicializar la variable**.







Buenas prácticas
Orden

- **Documentación:**
  - ✓ Para reducir la cantidad de comentarios, podemos utilizar nombres descriptivos en las variables.
  - ✓ Utilice oraciones completas al escribir comentarios. Los comentarios deben aclarar el código, no añadir ambigüedad.
  - ✓ Utilice los comentarios para explicar la intención del código. No deben servir como traducciones en línea del código.
  - ✓ Al comienzo de cada rutina, es útil proporcionar comentarios que indiquen el propósito, las suposiciones y las limitaciones de la rutina



Buenas prácticas



Mala práctica

```
Algoritmo Calculo_de_poliza
    constantes
        impuesto = 0.07      //Esta es la constante impuesto
    variables
        real precio          //Esta es la variable precio
        entero número        //Esta es la variable número
        cadena nombre        //Esta es la variable nombre

INICIO
    .....
FIN
```





Buenas prácticas



Ejemplo

```
/**
 * Update the specified resource in storage.
 *
 * @param  Request  $request
 * @return Response
 */
public function update(Request $request )
{
    // Trayendo los datos del usuario iniciado
    $persona_id= Auth::user()->persona_id;
    $personaDato = Persona::where('id', $persona_id)->get()->toArray();
    $data =  $request->input('data');
    $generales = $data["general"];
    $gfoot="";
    $generales["usuario_cotizador"] = $personaDato[0]["nombre_completo"];

    $dateCreado=date_create($generales["fecha"]);
    //dd($generales["fecha"]);
    $generales["fecha"]=date_format($dateCreado,"Y-m-d H:i");

    $arrfecha = explode("-",$generales["fecha"]);
  //* Para el nuevo formato revision 5
    $generales["year"]=$arrfecha[0];
    $generales["month"]=$arrfecha[1];
    $day=explode(" ", $arrfecha[2]);
    $generales["day"] = $day[0];

    //end Para el nuevo formato revision 5
    $generales["nombreJefeLab"]="Aun por definir";
    if(isset($data["Tnorma"]))
    {
```

Comentarios de introducción para la función.

Comentarios por bloque.

Comentarios innecesarios.

Comentarios de fin de bloque.







- **Para la próxima clase:**
  - **Control de Versiones**
    - **¿Qué es?**
    - **Para qué funcionan**
    - **Ejemplo de herramientas**





@utpfisc





@utpfisc



fisc@utp.ac.pa





www.fisc.utp.ac.pa

